

APPENDIX A

```
/*
** Oki Dev Program - Written by Michael Montgomery 4/10/97
** Copyright © 1997 Schlumberger Austin Products Center,
** Schlumberger Technologies, Austin, Texas, USA.
5  ** All Rights Reserved.
**
** This program was written to demonstrate the display request command
** to control the OKI display. This program monitors the keypad,
** and toggles the segment corresponding to the two keys entered.
10 ** The first key enters the row number (byte to be changed), and
** the second key enters the column number (bit to be changed).
**
** Pressing the Cancel key, immediately followed by Unlock key,
** cancels autoexecution of this program.
15 **
** Any other keypress is ignored.
**
*/

20 public class OkiDev {

    public static void main(String args[]) {

        // Send back the Answer To Reset (ATR)
25     _OS.SendATR();

        // Allocate command buffers
        byte[] keyscanbuffer = new byte[OkiDevConst.KEY_SCAN_CMD_LENGTH];
        byte[] keydatabuffer = new byte[OkiDevConst.KEY_DATA_CMD_LENGTH];
30     byte[] dispmapbuffer = new byte[OkiDevConst.DISP_MAP_CMD_LENGTH];

        // Allocate receive buffers
        byte[] receiveddatabuffer = new byte[OkiDevConst.RECEIVED_DATA_LENGTH];

35     // Build display command buffers
        dispmapbuffer[0] = OkiDevConst.ISO_ESCAPE;
        dispmapbuffer[1] = OkiDevConst.INS_LCD_DISPLAY;
        dispmapbuffer[2] = OkiDevConst.DISPLAY_SEGMENT_MAP;
        dispmapbuffer[3] = OkiDevConst.DISP_MAP_DATA_LENGTH;
40     dispmapbuffer[4] = (byte)0xFF;
        dispmapbuffer[5] = (byte)0xFF;
        dispmapbuffer[6] = (byte)0xFF;
        dispmapbuffer[7] = (byte)0xFF;
        dispmapbuffer[8] = (byte)0xFF;
45     dispmapbuffer[9] = (byte)0xFF;
        dispmapbuffer[10] = (byte)0xFF;
        dispmapbuffer[11] = (byte)0xFF;
        dispmapbuffer[12] = (byte)0xFF;
        dispmapbuffer[13] = (byte)0xFF;
50     dispmapbuffer[14] = (byte)0xFF;
        dispmapbuffer[15] = (byte)0xFF;
```

```

    dispmapbuffer[16] = (byte)0xFF;
    dispmapbuffer[17] = (byte)0xFF;
    dispmapbuffer[18] = (byte)0xFF;
    dispmapbuffer[19] = (byte)0xFF;

5    // Build key pad command buffers
    keyscanbuffer[0] = OkiDevConst.ISO_ESCAPE;
    keyscanbuffer[1] = OkiDevConst.INS_KEY_SCAN;
    keyscanbuffer[2] = OkiDevConst.PARAM_UNUSED;
10    keyscanbuffer[3] = OkiDevConst.PARAM_UNUSED;

    keydatabuffer[0] = OkiDevConst.ISO_ESCAPE;
    keydatabuffer[1] = OkiDevConst.INS_KEY_DATA;
    keydatabuffer[2] = OkiDevConst.PARAM_UNUSED;
15    keydatabuffer[3] = OkiDevConst.PARAM_UNUSED;

    // Initialize to send key scan command
    receiveddatabuffer[0] = 1;
    byte inputkey;

20    // Initialize unlock sequence: if cancel and unlock keys are pressed sequentially,
    // the card cancels automatic execution of this test program. To continue use
    // of this test program after this requires reselection of this program as
    // auto-execute in the development environment. This provision is made to
    // (hopefully) permit reuse and redownloading of this card in the event that
25    // either the test program has a bug, or the test program is not longer needed
    // and the card can be reused for another purpose.

    boolean cancelpressed = false;

30    // Initialize bit masks - selects appropriate bit to toggle
    byte[] mask = new byte[8];
    mask[0] = (byte)0x01;
    mask[1] = (byte)0x02;
    mask[2] = (byte)0x04;
35    mask[3] = (byte)0x08;
    mask[4] = (byte)0x10;
    mask[5] = (byte)0x20;
    mask[6] = (byte)0x40;
40    mask[7] = (byte)0x80;

    boolean firstkey = true;
    byte firstvalue = 0;

45    // Light all segments
    _OS.SendMessage(dispmapbuffer,OkiDevConst.DISP_MAP_CMD_LENGTH);
    _OS.GetMessage(receiveddatabuffer,(byte)0x02,OkiDevConst.ACK_CODE); // Ignore status reply

    boolean getmoredata = true;

50    //Main Loop (do forever)
    do {
        // Set up to buffer keystrokes
        if (getmoredata)
65    {
            _OS.SendMessage(keyscanbuffer,OkiDevConst.KEY_SCAN_CMD_LENGTH);

```

```

        _OS.GetMessage(receiveddatabuffer,(byte)0x02,OkiDevConst.ACK_CODE);
        getmoredata = false;
    }

5    // Get control keystroke (ignore all but first)
    _OS.SendMessage(keydatabuffer,OkiDevConst.KEY_DATA_CMD_LENGTH);

    _OS.GetMessage(receiveddatabuffer,OkiDevConst.RECEIVED_DATA_LENGTH,OkiDevConst.ACK_CODE);

10    // Select buffer to display

    if (receiveddatabuffer[0] != 0)
    {
        getmoredata = true;
        inputkey = receiveddatabuffer[1];
        if (inputkey == OkiDevConst.KEY_CODE_CANCEL)
        {
            cancelpressed = true;
        }
        else if (inputkey == OkiDevConst.KEY_CODE_UNLOCK)
        {
            if (cancelpressed) _OS.Execute((short)0, (byte)0);
        }
        else
        {
            cancelpressed = false;
            if (firstkey)
            {
                if ((inputkey < 0x10) && (inputkey >= 0))
                {
                    firstvalue = inputkey;
                    firstkey = false;
                }
            }
            else
            {
                firstkey = true;
                if ((inputkey < 8) && (inputkey >= 0))
                {
                    // Toggle display segment specified
                    dispmapbuffer[5 + firstvalue] ^= mask[inputkey];

                    // Display current map buffer

45    _OS.SendMessage(dispmapbuffer,OkiDevConst.DISP_MAP_CMD_LENGTH);
        _OS.GetMessage(receiveddatabuffer,(byte)0x02,OkiDevConst.ACK_CODE);
        }
    }
}

50    }
    }
    while (true);
}

55    public interface OkiDevConst{

```

// Constants used throughout the program

static final byte DISP_MAP_DATA_LENGTH = (byte)16;

static final byte DISP_MAP_CMD_LENGTH = DISP_MAP_DATA_LENGTH + (byte)4;

5 static final byte KEY_SCAN_CMD_LENGTH = (byte)4;

static final byte KEY_DATA_CMD_LENGTH = (byte)4;

static final byte ISO_ESCAPE = (byte)0xD0;

static final byte INS_LCD_DISPLAY = (byte)0xE0;

10 static final byte INS_KEY_SCAN = (byte)0xE1;

static final byte INS_KEY_DATA = (byte)0xE3;

static final byte PARAM_UNUSED = (byte)0x00;

static final byte DISPLAY_FIXED_POINT = (byte)0x01;

static final byte DISPLAY_HEXADecimal = (byte)0x02;

15 static final byte DISPLAY_SEGMENT_MAP = (byte)0x03;

static final byte KEY_CODE_CANCEL = (byte)0xF2;

static final byte KEY_CODE_UNLOCK = (byte)0xF1;

20 static final byte RECEIVED_DATA_LENGTH = (byte)3;

static final byte ACK_CODE = (byte)0;

}